

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of)	MAIL STOP
Jean-Jacques Vandewalle et al.)	APPEAL BRIEF - PATENTS
Application No.: 10/665,905)	Group Art Unit: 2442
Filed: September 15, 2003)	Examiner: JASON D. RECEK
For: METHOD AND MEANS FOR)	Appeal No.: _____
MANAGING COMMUNICATIONS)	
BETWEEN LOCAL AND)	
REMOTE OBJECTS IN AN)	
OBJECT ORIENTED CLIENT)	
SERVER SYSTEM IN WHICH A)	
CLIENT APPLICATION INVOKES)	
A LOCAL OBJECT AS A PROXY)	
FOR A REMOTE ..)	

APPEAL BRIEF

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This appeal is from the decision of the Primary Examiner dated November 25, 2009 finally rejecting claims 1-9 and 14-31, which are reproduced as the Claims Appendix of this brief.

☒ Charge ☐ \$ 270 ☒ \$ 540 to Credit Card.

The Commissioner is hereby authorized to charge any appropriate fees under 37 C.F.R. §§ 1.17 and 41.20 that may be required by this paper, and to credit any overpayment, to Deposit Account No. 02-4800.

Table of Contents

I.	Real Party in Interest.....	2
II.	Related Appeals and Interferences	2
III.	Status of Claims	2
IV.	Status of Amendments	2
V.	Summary Claimed Subject Matter.....	2
VI.	Grounds of Rejection to be Reviewed on Appeal.....	7
VII.	Argument	7
VIII.	Claims Appendix	12
IX.	Evidence Appendix	13
X.	Related Proceedings Appendix.....	13

I. Real Party in Interest

GEMALTO SA, which is the successor in interest to GEMPLUS, the original assignee of the application, is the real party in interest.

II. Related Appeals and Interferences

The Appellants' legal representative, or assignee, does not know of any other appeal, interferences or judiciary proceedings which will affect or be directly affected by or have bearing on the Board's decision in the pending appeal.

III. Status of Claims

The application contains claims 1-31. Claims 10-13 have been cancelled previously. To simply issues for the appeal, claims 14-28, 30 and 31 are being cancelled in a supplemental Amendment filed concurrently with the Appeal Brief. Accordingly, claims 1-9 and 29 are currently pending. Claims 1-9 and 29 stand finally rejected, and are being appealed.

IV. Status of Amendments

A supplemental Amendment is being filed concurrently herewith, cancelling claims 14-28, 30 and 31. For purposes of this appeal, it is assumed that this Amendment will be entered, since it complies with the requirements of 37 C.F.R. § 41.33(b)(1).

No other amendments were filed subsequent to the final Office Action dated November 25, 2009.

V. Summary of Claimed Subject Matter

The claimed subject matter is directed to a method for managing information exchanges among communicating objects in an object-oriented client server system.

Pursuant to 37 C.F.R. §41.37(1)(c)(v), the subject matter of independent claims 1, 4 and 6 is cross-referenced to the specification and/or drawing figures in the following table. The following table is not to be construed as a representation that the portions of the disclosure identified below constitute the sole basis for support for the claimed subject matter.

Claim	Disclosure
1. A method for managing information exchanges among communicating objects in an object-oriented client server system, said system including first and second object-oriented virtual machines running on counterpart first and second computers in respective server and client roles, and a communication path connection between said computers, said server virtual machine having a run-time environment, the method comprising:	The paragraph bridging pages 6 and 7; and Figs. 3C and 5
(a) generating a local object at the client machine based upon interface definition of a remote object resident at the server machine, said local object executable as a proxy to the remote object; said server machine residing in a smart device; and said client machine having access to the smart device via a smart device reader;	The paragraph bridging pages 6 and 7; the paragraph bridging pages 12 and 13; the paragraph bridging pages 14 and 15; and Fig. 5, reference numerals 501-515
(b) referencing the local object by an application executing at the client machine and causing the local object to marshal parameters;	The paragraph bridging pages 6 and 7; the paragraph bridging pages 12 and 13; the paragraph bridging pages 14 and 15; and Fig. 5, reference numeral 503
(c) sending a process level call request by direct method invocation to the run-time environment of the server machine;	The paragraph bridging pages 6 and 7; the paragraph bridging pages 12 and 13; the paragraph bridging pages 14 and 15; and Fig. 5,

	reference numerals 507 and 509
(d) responsive to receipt of said request by the server machine's run-time environment, said run-time environment causing the parameters in the request to become unmarshaled, said remote object to be executed, and the results of the execution to be marshaled;	The paragraph bridging pages 6 and 7; the paragraph bridging pages 12 and 13; the paragraph bridging pages 14 and 15; and Fig. 5, reference numerals 513 and 515
(e) sending a process level return to the client machine as a reply; and	The paragraph bridging pages 6 and 7; the paragraph bridging pages 12 and 13; the paragraph bridging pages 14 and 15; and Fig. 5, reference numerals 509 and 513
(f) responsive to said reply, unmarshaling the results from said reply by the local object at the client machine.	The paragraph bridging pages 6 and 7; the paragraph bridging pages 12 and 13; the paragraph bridging pages 14 and 15; and Fig. 5, reference numerals 503 and 507
4. A method for managing information exchanges between an application executing at a object-oriented virtual machine operable as a client and a remote object resident at another object-oriented virtual machine operable as a server, said server machine having a run-time environment, said client and server having a communication path connection there-between, said communication path connection being operable under a process for originating and sending byte level messages therebetween, comprising:	The paragraph bridging pages 6 and 7; and Figs. 3C and 5

<p>(a) providing a local object resident at the client machine executable as a proxy stub to the remote object resident at the server machine and providing a description of the remote object to enable said run-time environment to also operate as a stub, said server machine residing in a smart device; and said client machine having access to the smart device via a smart device reader; wherein the local object is generated based upon interface definition of a remote object resident at the server machine;</p>	<p>The paragraph bridging pages 6 and 7; the paragraph bridging pages 12 and 13; the paragraph bridging pages 14 and 15; and Fig. 5, reference numerals 501-515</p>
<p>(b) responsive to a client application call to the local object, marshaling parameters and causing a process level call request to be sent to the run-time environment of the server machine, said sending of the request further including mapping said process level call request into counterpart byte string level messages and transmitting said messages to the server machine;</p>	<p>The paragraph bridging pages 6 and 7; the paragraph bridging pages 12 and 13; the paragraph bridging pages 14 and 15; and Fig. 5, reference numerals 503, 507 and 509</p>
<p>(c) responsive to receipt of said request messages by the server machine's run-time environment, mapping said messages into a process level call request, unmarshaling the parameters, invoking and executing the remote object, marshaling the results, forming a process level reply, mapping said reply into string byte messages, and transmitting said reply messages to the client machine; and</p>	<p>The paragraph bridging pages 6 and 7; the paragraph bridging pages 12 and 13; the paragraph bridging pages 14 and 15; and Fig. 5, reference numerals 507, 509, 513 and 515</p>
<p>(d) responsive to the reply messages by the proxy at the client machine, mapping said reply messages into a process level reply, and unmarshaling the results.</p>	<p>The paragraph bridging pages 6 and 7; the paragraph bridging pages 12 and 13; the paragraph bridging pages 14 and 15; and Fig. 5, reference numerals 501 and 503</p>
<p>6. An article of manufacture comprising a machine readable memory having stored therein a plurality of processor executable control program steps for managing information exchanges among communicating objects in an object-oriented client</p>	<p>The paragraph bridging pages 6 and 7; and Figs. 3C and 5</p>

server system, said system including first and second object-oriented virtual machines running on counterpart first and second computers in respective server and client roles, and a communication path connection between said computers, said server virtual machine having a run-time environment, said control program steps including:	
(a) a control program step for generating a local object at the client machine executable as a proxy to a remote object resident at the server machine, said server machine residing in a smart device; and said client machine having access to the smart device via a smart device reader; wherein the local object is generated based upon interface definition of a remote object resident at the server machine;	The paragraph bridging pages 6 and 7; the paragraph bridging pages 12 and 13; the paragraph bridging pages 14 and 15; and Fig. 5, reference numerals 501-515
(b) a control program step for referencing the local object by an application executing at the client machine and causing the local object to marshal parameters;	The paragraph bridging pages 6 and 7; the paragraph bridging pages 12 and 13; the paragraph bridging pages 14 and 15; and Fig. 5, reference numerals 503, 507 and 509
(c) a control program step for transmitting a process level call request to the server machine's run-time environment;	The paragraph bridging pages 6 and 7; the paragraph bridging pages 12 and 13; the paragraph bridging pages 14 and 15; and Fig. 5, reference numerals 507 and 509
(d) a control program step responsive to receipt of said request by the server machine's run-time environment, to cause said run-time environment to unmarshal the parameters in the request, execute said remote object, marshal the results of the execution, and send a process level return to the client machine; and	The paragraph bridging pages 6 and 7; the paragraph bridging pages 12 and 13; the paragraph bridging pages 14 and 15; and Fig. 5, reference numerals 513 and 515

(e) a control program step responsive to said return to cause said local object to unmarshal the results from said reply.	The paragraph bridging pages 6 and 7; the paragraph bridging pages 12 and 13; the paragraph bridging pages 14 and 15; and Fig. 5, reference numeral 503

VI. Grounds of Rejection to be Reviewed on Appeal

A. Claims 1-3, 6, 7 and 29 are rejected under 35 U.S.C. §103(a), on the basis of the Java Remote Method Invocation Specification published February 10, 1997 by Sun Microsystems (identified in the Office Action as "JDOM"), in view of the Jones et al. patent (U.S. 6,557,032, hereinafter "Jones") and the Dievendorff et al. patent (U.S. 6,425,017, hereinafter "Dievendorff").

B. Claims 4, 5, 8 and 9 are rejected under 35 U.S.C. §103(a), on the basis of JDOM and Jones, in view of alleged prior art (hereinafter, "APA"), and in further view of Dievendorff.

VII. Argument

1 Appellants' Exemplary Embodiments

Appellants' exemplary embodiments relate to managing communications between local and remote objects in an object oriented client server system in which a client application residing at client, such as a terminal, invokes a local object as a proxy for a remote object on a server residing in a smart device, e.g., a smart card.

In conventional Java cards, a protocol specification describes a limited version of the Java language and platform tuned to smart cards, and thus, existing

client applications at terminals are limited by such protocol specification. As such, the terminals can only communicate with the conventional Java cards through already defined Application Program Data Units (APDUs).

According to Appellant's exemplary embodiments, a local object at a client machine, e.g., a terminal, is generated as a proxy to a remote object resident at a server machine, e.g., a smart device. The local object is called by an application executing at the client machine, and the local object is caused to marshal parameters and send a process level call request to the server machine. Next, the server machine's run time environment causes the parameters in the request to become unmarshaled, and the remote object to be executed. The results of the execution are marshaled, and a process level return is sent to the client machine. Then, at the client machine, the results in the process level return are unmarshaled.

The use of a local object at the terminal as proxy to a remote object resident at the smart device according to Appellant's exemplary embodiments provides a more flexible communication between the terminal and the smart device, compared to a conventional Java card. As such, the communication between the terminal and the smart device according to Appellant's exemplary embodiments is not limited to already defined APDUs.

2 Claims 1-3, 6, 7 and 29

Claim 1 recites a method for managing information exchanges among communicating objects in an object-oriented client server system. The method comprises, among other steps:

generating a local object at the client machine based upon interface definition of a remote object resident at the server machine, said local object executable as a proxy to the remote

object; said server machine residing in a smart device; and said client machine having access to the smart device via a smart device reader.

The Official Action dated November 25, 2009 acknowledges that JDOM does not disclose “said server machine residing in a smart device; and said client machine having access to the smart device via a smart device reader”, and relies upon col. 3, lines 16-26 and lines 50-58 of the Jones patent as allegedly disclosing such features. See the Official Action: page 6, the first paragraph. Appellants respectfully disagree.

The Jones patent discloses a distributed data processing system, such as a retail store customer transaction network. Referring to Fig. 1 of the Jones patent, a distributed data processing system includes a terminal 1 for handling customer transactions in a retail environment, enabling each transaction to be initiated by the insertion of a smart card 3 which is read and written via an interface 4 to provide and receive data relating to the transaction. Each transaction is controlled by a generic application processor 4 in the terminal 1 in which a Java language interpreter is established and various application objects associated with a transaction are instantiated. An application library 5 stores objects relating to the transactions in the terminal 1, and is connected to the processor 4. Also connected to the processor 4 is a user interface library 6, which stores objects to establish the proper interface between the input/output equipment 2 and the types of smart card 3 likely to be presented to the terminal 1.

In the Official Action, the Examiner refers to col. 3, lines 16-26 of Jones as disclosing servers that perform using smart cards. However, a server that uses a smart card to perform a function is not the same as a server that resides on a smart device, as recited in claim 1. The cited passage of the Jones patent refers to the servers 9 and 10 that are illustrated in Figure 1. As can be seen, these servers are

remote from the smart card 3 connected to the terminal 1. At column 3, lines 39-44, the patent states:

[A] situation could arise for example where a card is presented of a type not normally handled by the terminal but which is commonly used in an application elsewhere in the network. Terminal 1 then initiates a request for the appropriate object to be made accessible through server 9 or 10.

From the foregoing it is believed to be clear that the servers described at column 3, lines 16-26 are quite distinct from the smart card 3. As such, these servers can not be considered to correspond to, or otherwise suggest, the claimed "server machine residing in a smart device."

In the Advisory Action dated April 14, 2010, the Examiner states that the specification indicates that a smart card may be an open computing platform. Thus, the computing platform disclosed by Jones constitutes a smart device, if interpreted broadly. The Examiner asserts that since Jones discloses a server included in the computing platform, Jones discloses a server machine residing in a smart device.

Appellants submit that the Examiner's logic used in the above assertion is flawed. Just because a smart card may be an open computing platform, it does not mean that anything with an open computing platform is a smart card. Therefore, it is incorrect for the Examiner to assert that the server disclosed in Jones corresponds to the claimed smart device.

Furthermore, Appellants submit that claim language should not be interpreted in a vacuum. Instead, claim language should be read in light of the specification, as it would be interpreted by one of ordinary skill in the art.

The present specification discloses methods effectuating exchanges among remote objects residing on server machines that include chip or smart card based virtual machines and counterparts, and local objects at client machines that include

terminals or the like. According to the disclosure in the present specification, using a local object at a terminal as proxy to a remote object resident at a server machine that includes chip or smart card based virtual machines or counterparts provides a more flexible communication between the terminal and the server machines, compared to a conventional Java card. For example, unlike communication with a conventional Java card, the communication to a server machine that includes chip or smart card based virtual machines or counterparts according to Appellants' exemplary embodiments is not limited to predefined APDUs.

Therefore, according to the present specification, Appellants' exemplary embodiments provide an easy access or usage of objects residing on the server machine that includes chip or smart card based virtual machines or counterparts. As such, when interpreted in light of the specification, the term "smart device" as used in recitation "server machine residing in a smart device" should be considered as a server machine that includes a chip or smart card based virtual machine, or similar such counterparts.

A server that is remote from a smart card and its connected terminal as disclosed in the Jones patent does not have much resemblance to chip or smart card based virtual machines or counterparts. For example, accessing the server as disclosed in Jones does not require the use of APDUs. In addition, claim 1 recites that access to the smart device is via a smart device reader. For example, a smart card is read by a smart card reader. Access to the server disclosed in the Jones patent does not require a "server reader." As such, the server as disclosed in Jones should not be considered as corresponding to the claimed smart device.

As explained above, neither the smart card, nor the server, as disclosed in the Jones patent corresponds to the claimed smart device. Therefore, the Jones patent

does not disclose “said server machine residing in a smart device; and said client machine having access to the smart device via a smart device reader,” as recited in claim 1. As such, Jones fails to remedy the deficiencies of JDOM.

Dievendorff relates to an object runtime architecture that allows method invocations to be made on either a synchronous, real-time basis or a queued basis using the normal call semantics of an object model. The Dievendorff patent does not have any disclosure on information exchanges among communicating objects involving a smart device. Therefore, the Dievendorff patent does not remedy the deficiencies of the JDOM article and the Jones patent.

Consequently, any logical combination of the teachings of JDOM, Jones and Dievendorff does not suggest the subject matter of claim 1, and therefore claim 1 is patentable. Claims 2, 3, and 29 are patentable at least because of their dependency from claim 1. Claims 6 and 7 are patentable at least because they include distinguishing features similar to those of claim 1.

3 Claims 4, 5, 8 and 9

The alleged APA does not disclose that a server machine, at which a remote object is located, resides in a smart device, as described in claim 1. Therefore, APA does not remedy the above-noted deficiencies of JDOM, Jones and Dievendorff. Accordingly, claims 4, 5, 8 and 9 are patentable.

VIII. Claims Appendix

See attached Claims Appendix for a copy of the claims involved in the appeal.

IX. Evidence Appendix

None

X. Related Proceedings Appendix

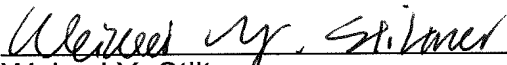
None

Respectfully submitted,

BUCHANAN INGERSOLL & ROONEY PC

Date July 26, 2010

By:


Weiwei Y. Stiltner
Registration No. 62979

Customer No. 21839
703 836 6620

VIII. CLAIMS APPENDIX

The Appealed Claims

1. A method for managing information exchanges among communicating objects in an object-oriented client server system, said system including first and second object-oriented virtual machines running on counterpart first and second computers in respective server and client roles, and a communication path connection between said computers, said server virtual machine having a run-time environment, the method comprising:

(a) generating a local object at the client machine based upon interface definition of a remote object resident at the server machine, said local object executable as a proxy to the remote object; said server machine residing in a smart device; and said client machine having access to the smart device via a smart device reader;

(b) referencing the local object by an application executing at the client machine and causing the local object to marshal parameters;

(c) sending a process level call request by direct method invocation to the run-time environment of the server machine;

(d) responsive to receipt of said request by the server machine's run-time environment, said run-time environment causing the parameters in the request to become unmarshaled, said remote object to be executed, and the results of the execution to be marshaled;

(e) sending a process level return to the client machine as a reply; and

(f) responsive to said reply, unmarshaling the results from said reply by the local object at the client machine.

2. The method according to claim 1, wherein plural process call level requests and replies are generated in an alternating manner.

3. The method according to claim 1, wherein the local object when operating as a proxy at the client machine and the run-time environment when operating at the server machine perform respectively as stubs.

4. A method for managing information exchanges between an application executing at a object-oriented virtual machine operable as a client and a remote object resident at another object-oriented virtual machine operable as a server, said server machine having a run-time environment, said client and server having a communication path connection there-between, said communication path connection being operable under a process for originating and sending byte level messages therebetween, comprising:

(a) providing a local object resident at the client machine executable as a proxy stub to the remote object resident at the server machine and providing a description of the remote object to enable said run-time environment to also operate as a stub, said server machine residing in a smart device; and said client machine having access to the smart device via a smart device reader; wherein the local object is generated based upon interface definition of a remote object resident at the server machine;

(b) responsive to a client application call to the local object, marshaling parameters and causing a process level call request to be sent to the run-time environment of the server machine, said sending of the request further including

mapping said process level call request into counterpart byte string level messages and transmitting said messages to the server machine;

(c) responsive to receipt of said request messages by the server machine's run-time environment, mapping said messages into a process level call request, unmarshaling the parameters, invoking and executing the remote object, marshaling the results, forming a process level reply, mapping said reply into string byte messages, and transmitting said reply messages to the client machine; and

(d) responsive to the reply messages by the proxy at the client machine, mapping said reply messages into a process level reply, and unmarshaling the results.

5. The method according to claim 4, wherein said object-oriented virtual machines include Java virtual machines, and further wherein the remote object is an applet, and the local object is an interface description.

6. An article of manufacture comprising a machine readable memory having stored therein a plurality of processor executable control program steps for managing information exchanges among communicating objects in an object-oriented client server system, said system including first and second object-oriented virtual machines running on counterpart first and second computers in respective server and client roles, and a communication path connection between said computers, said server virtual machine having a run-time environment, said control program steps including:

(a) a control program step for generating a local object at the client machine executable as a proxy to a remote object resident at the server machine, said server

machine residing in a smart device; and said client machine having access to the smart device via a smart device reader; wherein the local object is generated based upon interface definition of a remote object resident at the server machine;

(b) a control program step for referencing the local object by an application executing at the client machine and causing the local object to marshal parameters;

(c) a control program step for transmitting a process level call request to the server machine's run-time environment;

(d) a control program step responsive to receipt of said request by the server machine's run-time environment, to cause said run-time environment to unmarshal the parameters in the request, execute said remote object, marshal the results of the execution, and send a process level return to the client machine; and

(e) a control program step responsive to said return to cause said local object to unmarshal the results from said reply.

7. The method according to claim 1, wherein said client machine accesses the smart device with communication protocols specified according to International Standards Organization specification 7816-4.

8. The method according to claim 7, wherein said client machine obtains access to the smart device via a command Application Program Data Unit.

9. The method according to claim 1, wherein said reply is formatted into an Application Program Data Unit response.

29. The method of claim 1, wherein the smart device comprises a smart card.

IX. EVIDENCE APPENDIX

None

X. RELATED PROCEEDINGS APPENDIX

None